```
%NETBP Uses backpropagation to train a network
%% %%%%%%% DATA %%%%%%%%%%%

clear all
%%%Problem 3: You can vary the width (line 14) and Niter (line 19)
x1 =[1.888,-1.893,-1.856,-1.535,-1.395,-0.583,↙
-0.0986,0.2939,1.505,1.772,1.888,1.093,0.368,0.044,-0.075,-0.75234,-1.44,-1,0,1,1.5,↙
-1.5,0,0,2,2.145,3.235,-2.235,-2.762,0.85,-2,-2.525,-1.543,2.097,-0.525,-2.051,2.523,↙
-0.5,0.5,1];
x2=[-0.203,-0.250,-0.800, -1.795, -2.15, -2.845, -3.104, -3.015, -1.855, -1.249, -0.2035,↙
0.9087, 0.741, 0.111, 0.189, 0.881, 0.759, -1, -2,-1, -3.95, -3.68, 0.835, -4.523, 3.215,↙
1, 3.1252, -1.235, -3.12, 1.580, 1.8, 2.151, 2.656, -1.459, -3.862, 1.533, 0.579, 1.5,↙
1.5,-3.230];
x1 = (x1-min(x1))/(max(x1)-min(x1));%normalize
x2 = (x2-min(x2))/(max(x2)-min(x2));%normalize
y = [ones(1,20) zeros(1,20); zeros(1,20) ones(1,20)];

%% Initialize weights and biases
rng(5000);
width = 90;
W2 = 1/sqrt(width)*randn(width,2); W3 = 1/sqrt(width)*randn(width,width); W4 = 1/sqrt↙
(width)*randn(width,width); W5 = 0.5*randn(2,width);
b2 = 1/sqrt(width)*randn(width,1); b3 = 1/sqrt(width)*randn(width,1); b4 = 1/sqrt(width)↙
*randn(width,1); b5 = 0.5*randn(2,1);
eta = 0.08  ; % learning rate
Niter = 3e7; % number of SG iterations
savecost = zeros(Niter,1); % value of cost function at each iteration

%% Training
for counter = 1:Niter
    % choose a training point at random
    k = randi(40);
    x = [x1(k); x2(k)];

    % Forward pass
    a2 = activate(x,W2,b2);
    a3 = activate(a2,W3,b3);
    a4 = activate(a3,W4,b4);
    a5 = activate(a4,W5,b5);

    % Backward pass
    delta5 = a5.*(1-a5).*(a5-y(:,k));
    delta4 = a4.*(1-a4).*(W5'*delta5);
    delta3 = a3.*(1-a3).*(W4'*delta4);
    delta2 = a2.*(1-a2).*(W3'*delta3);

    % Gradient updates
    W2 = W2 - eta*delta2*x';
    W3 = W3 - eta*delta3*a2';
    W4 = W4 - eta*delta4*a3';
    W5 = W5 - eta*delta5*a4';
    b2 = b2 - eta*delta2;
    b3 = b3 - eta*delta3;
    b4 = b4 - eta*delta4;
    b5 = b5 - eta*delta5;

    % Monitor progress
    newcost = cost(W2,W3,W4,W5,b2,b3,b4,b5,x1,x2,y); % display cost to screen
    savecost(counter) = newcost;

    % Print every 100 iterations
    if mod(counter, 1000) == 1
        fprintf('Iteration: %d, Cost: %f\n', counter, newcost);
    end

end

%% Plotting
% Show decay of cost function
save costvec
semilogy([1:1e3:Niter],savecost(1:1e3:Niter))
xlabel('Iteration')
ylabel('cost')

%%Plotting
close all
hold on

%testing points
for k = 1:5000
    % choose a training point at random

    x = rand(2,1); %random test point

    % Forward pass
    a2 = activate(x,W2,b2);
    a3 = activate(a2,W3,b3);
    a4 = activate(a3,W4,b4);
    a5 = activate(a4,W5,b5);


    if a5(1)>a5(2)
        plot(x(1),x(2),'k.'); xlim([0 1]); ylim([0 1]);
    end
end

for k = 1:40
    % choose a training point

    x = [x1(k); x2(k)];

    if y(1,k)>y(2,k)
        plot(x(1),x(2),'bo', 'MarkerFaceColor', 'b'); xlim([0 1]); ylim([0 1]);
    elseif y(1,k)<y(2,k)
        plot(x(1),x(2),'ro', 'MarkerFaceColor', 'r'); xlim([0 1]); ylim([0 1]);
    end

end

function costval = cost(W2,W3,W4,W5,b2,b3,b4,b5,x1,x2,y)
    costvec = zeros(40,1);
    for i = 1:40
        x =[x1(i);x2(i)];
        a2 = activate(x,W2,b2);
        a3 = activate(a2,W3,b3);
        a4 = activate(a3,W4,b4);
        a5 = activate(a4,W5,b5);
        costvec(i) = norm(y(:,i) - a5,2);
    end
    costval = 1/40*1/2*norm(costvec,2)^2;
end

function y = activate(x,W,b)
    %ACTIVATE Evaluates sigmoid function.
    %
    % x is the input vector, y is the output vector
    % W contains the weights, b contains the shifts
    %
    % The ith component of y is activate((Wx+b)_i)
    % where activate(z) = 1/(1+exp(-z))
    y = 1./(1+exp(-(W*x+b)));
end
```